Secure Cloud Connectivity for Scientific Applications

Osmani, Lirim

2018-07

Osmani , L , Toor , S , Komu , M , Kortelainen , M J , Linden , T , White , J , Khan , R , Eerola , P & Tarkoma , S 2018 , ' Secure Cloud Connectivity for Scientific Applications ' , IEEE Transactions on Services Computing , vol. 11 , no. 4 , pp. 658-670 . https://doi.org/10.1109/TSC.2015.246929

http://hdl.handle.net/10138/327068
https://doi.org/10.1109/TSC.2015.2469292

unspecified
acceptedVersion

Secure Cloud Connectivity for Scientific Applications

Lirim Osmani, Salman Toor, Miika Komu, Matti J. Kortelainen,
Tomas Lindén, John White, Rasib Khan, Paula Eerola, Sasu Tarkoma

Abstract—Cloud computing improves utilization and flexibility in allocating computing resources while reducing the infrastructural costs. However, in many cases cloud technology is still proprietary and tainted by security issues rooted in the multi-user and hybrid cloud environment. A lack of secure connectivity in a hybrid cloud environment hinders the adaptation of clouds by scientific communities that require scaling-out of the local infrastructure using publicly available resources for large-scale experiments. In this article, we present a case study of the DII-HEP secure cloud infrastructure and propose an approach to securely scale-out a private cloud deployment to public clouds in order to support hybrid cloud scenarios. A challenge in such scenarios is that cloud vendors may offer varying and possibly incompatible ways to isolate and interconnect virtual machines located in different cloud networks. Our approach is tenant driven in the sense that the tenant provides its connectivity mechanism. We provide a qualitative and quantitative analysis of a number of alternatives to solve this problem. We have chosen one of the standardized alternatives, Host Identity Protocol, for further experimentation in a production system because it supports legacy applications in a topologically-independent and secure way.

Index Terms—Cloud Computing, Security, VPN, Scientific Applications, Grid, Cluster

1 Introduction

Computational clouds are not new to the scientific computing community. Over the past few years this community has very well understood the benefits of this concept by transforming their applications to virtualized service offerings at a rapid pace. The benefits include easier and faster deployment of scientific applications, and the use of public cloud offerings for rapidly scaling out services.

A hybrid cloud employs both a private and public cloud for rapid deployment of services. This is crucial for computationally demanding applications which require expedited scalability at global scale. As such hybrid clouds represent an opportunity for scientific applications as well. Organizations temporarily scaling-out their infrastructure from a private to a public cloud during peak hours, primarily have an economic advantage by eliminating any upfront cost for new hardware, and secondly reducing any administrative overhead with new resources introduced.

Scientific applications are inherently complex and often have dependencies on legacy components that are difficult to provide or maintain. Thus, existing scientific applications should be virtualized in a backward-compatible way. In addition, utilizing hybrid clouds represent additional challenges, such as interoperability between different cloud vendors. The interoperability aspects include portability of virtual machines, cross-vendor connectivity between virtual machines, and securing of computation, data and networks. In this paper, we touch "the tip of iceberg" and make some of the challenges more explicit, and show an approach for running virtualized scientific applications in a hybrid cloud setting. Our main contribution is in the design, implementation, evaluation and analysis of the proposed solution using a highly challenging scientific application that requires legacy components and secure connectivity in a distributed environment. The scientific application is a high energy physics grid software bundle that we have virtualized and ported to an OpenStack production environment.

The proposed security architecture is based on the already standardized Host Identity Protocol (HIP) [1]. However, instead of its typical client-to-server deployment model, we deploy it for server-to-server communications to minimize unnecessary deployment hurdles at the client side. To be more precise, we employ HIP only for intra and inter-cloud Virtual Machine (VM) communication to tackle some of the network security challenges related to hybrid clouds. The suggested approach can be incrementally deployed by the tenants independently of the cloud provider, in private or commercial cloud offering such as Amazon EC2[1].

This work has been conducted under the Datacenter Indirection Infrastructure for Secure HEP (High Energy Physics) Data Analysis (DII-HEP) project. The presented infrastructure delivers a production-ready cloud-based platform for Compact Muon Solenoid (CMS)[2] data analysis. The site *DII-HEP(CMS)* is part of the CMS collaboration and has been providing services for more than a year. HIP based security framework is a recent extension that allows us to scale-out the CMS computing infrastructure using public and community cloud resources.

1. *http://aws.amazon.com/ec2*
2. *http://cms.web.cern.ch*

The live status can be viewed from Advanced Resource Connector (ARC) site.

The presented empirical measurements help to quantify our approach. As practical results, we show that the proposed solution, HIP, offers a very similar performance as the de-facto TLS/SSL. Furthermore, the HIP-based security does not require any changes at the application level, and it is feasible to use HIP inside a cloud stack without intervening in the underlying complex network configuration.

Contributions: The key contributions of this paper include:

1) Applicability analysis of HIP for a computationally demanding scientific application;
2) Design and development of a system architecture for a secure cloud infrastructure and a scaling-out mechanism for the local infrastructures;
3) Experimental evaluation of the proposed architecture to evaluate the applicability of HIP for secure cloud communication for scientific computing.

The paper is organized as follows: Sections 2 and 3 describe the design constraints and proposed solution for enabling scientific applications on clouds; Section 4 examines the components used in deploying our cloud solution; System architecture is discussed in Section 5; Section 6 presents the architecture of hybrid clouds; Experimental setup and results are presented in Section 7 and 8 respectively, followed by a security review of the architecture in Section 9; Finally, Section 10 highlights our conclusions and future directions.

2 Design Requirements and Constraints

The ultimate goal of this work is to be able to run virtualized scientific applications in a hybrid cloud setting. In order to meet this goal, we divide it into smaller requirements that will be later referred by their number (marked as REQx). We argue and enumerate the requirements so that our qualitative contributions become more clear, and also try to set up a base line for other researchers to improve the state of the art.

As the first requirement, *the overall solution should be completely based on open-source software* (REQ1). This way, repeating of the experiments is easier and vendor lock-in can be avoided by the scientific community.

In order to virtualize a scientific application to the cloud, two different approaches could be taken: legacy or cloud native application virtialization. In the former approach, a legacy application is merely encapsulated into a virtual machine image to be executed by a hypervisor or in a Linux container in an Infrastructure as a Service (IaaS) type of environment. In the latter approach, an application is rewritten to use a Platform as a Service (PaaS) API and distributed using the platform provided by the cloud vendor. Existing scientific applications tend to be rather complex, and rewriting one is a tedious task. For this reason, we require the *virtualization approach to retain compatibility with legacy applications* (REQ2). Effectively, this limits us to employ hypervisor-based virtualization.

In addition to private clouds, we assume that scientific applications can be executed in a public cloud. Consequently, multi-tenancy becomes an issue. In other words, the computation, storage and network of different tenants should be isolated properly from each other. In the extreme case, a public cloud can operate [2] on encrypted data or be used as a storage for encrypted data [3], without the cloud vendor ever seeing the unencrypted data. However, such approaches incur a negative trade-off with performance and are not a bullet-proof solution for all cloud privacy issues [4]. In addition, such approaches are an overkill for our use case with the data from Large Hadron Collider (LHC), where the data merely needs to be isolated from the outsiders of the community. Thus, we require only *basic multi-tenancy support* from the underlying cloud platform (REQ3). This applies to computation, network and storage aspects occurring internally within a single cloud.

The underlying cloud platform should provide support for *scaling out* so that new compute nodes can be added (or removed) on demand (REQ4). Further, we assume that the platform supports dynamic transferring of computation. As the first requirement constrains the solution space to IaaS, this refers to *live migration of VMs within a single cloud* (REQ5). It should be noted that we do not require live migration support between hybrid clouds because it is not well supported in any cloud software, and typically requires custom solutions, but also both clouds have to run the same hypervisor.

Application level scale-out, fault tolerance and recovery in the orchestration framework is also required in order to build a robust service (REQ6). For instance, if one worker node fails, another one should be assigned to the workload to complete the processing. We assume also that the *users of the scientific application are authenticated and authorized in some way (REQ7).*

Next, we list a number of requirements related to network communication between individuals VMs. In hybrid cloud scenarios, mixing two different cloud deployments with potentially incompatible IP address namespaces further requires *an "elastic" addressing scheme across multiple clouds* so that VMs remain reachable to each other (REQ8). These connectivity issues are typically related to middleboxes (e.g. firewalls blocking connections and NAT devices with overlapping namespaces), IPv6 address compatibility and maintaining of transport-layer sessions during address changes [5]. As the connectivity passes through different, possibly incompatible security domains in the case of hybrid clouds, we argue that *connectivity between individual VMs should be secured* (REQ9). Related to this, a corollary of the first requirement is that *VM-to-VM connectivity and security related to hybrid clouds should be vendor agnostic* (REQ10) in order to avoid vendor lock-in. Finally, the complex nature of scientific applications basically mandates that the *VM-to-VM connectivity and security must work with unmodified legacy*

applications (REQ11).

3 Design Alternatives

Requirements from 1 to 6 are related to the cloud platforms. Popular open-source (REQ1) cloud platforms include OpenNebula, OpenStack, CloudStack and Eucalyptus [5]. All of them support virtualization of legacy applications using hypervisors such as KVM (REQ2). All of the platforms include basic security measures to support multi-tenant environments (REQ3). Similarly, basic support for scaling out (REQ4) and live migration of VMs is supported in all platforms (REQ5). We have chosen OpenStack for our experiments mostly due to its familiarity to us, but also its fast adoption rate.

Requirements 6 and 7 are related to the orchestration frameworks. In the context of this paper, we have to resort to ARC middleware due to its support in the CERN experiments. ARC middleware supports application level fault tolerance and recovery (REQ6), and authentication and authorization of the users (REQ7).

Table 1 shows a comparison of popular open-source solutions for requirements 8 – 11 based on on our earlier work [6, 7]. The Achilles heel in most of the protocols is that they fail to support mobility at both ends of the communication. Hence, the tradional Virtual Private Network (VPN)-like solutions, OpenVPN, StrongSWAN, MIPv4 from HP and UMIPv6, can only partially meet requirement 8 (elastic addressing scheme).

The remaining three protocols meet the criteria for all requirements. OpenLISP tunnels traffic between two or more networks based on LISP-capable routers and has been suggested to be used with virtualized grids in the context of "Intercloud"-architecture [5], albeit never actually realized by experimentation. While OpenLISP is a middlebox-based solution, the remaining two, iPoP and HIPL, are essentially end-to-end VPNs that require software installation at the end-hosts.

Software	REQ8	REQ9	REQ10	REQ11
OpenVPN	(✓)	✓	✓	✓
StrongSWAN	(✓)	✓	✓	✓
MIPv4/HP	(✓)	✓	✓	✓
UMIPv6	(✓)	✓	✓	✓
OpenLISP	✓	✓	✓	✓
iPoP	✓	✓	✓	✓
HIPL	✓	✓	✓	✓

TABLE 1: A comparison of open-source VPNs

Of last three protocols conforming to all requirements, we were inclined towards HIPL [8] in the production environment, even though we experimented with some others in order to compare their performance. Our reasoning was as follows: OpenLISP requires support in middleboxes, which can be problematic from the viewpoint of deployment and adoption. In contrast, HIP and iPoP can be installed only in the VMs that require it. Of the two remaining ones, the Host Identity Protocol for Linux (HIPL), has been standardized [1, 9], so we have chosen it for more extensive experimentation.

4 System Components

The building blocks of our solution are based on open source tools. This approach provides access to a vibrant community with already established ecosystem of tools and technical know-how. Finally, open source projects add functionality at a lot faster rate than proprietary projects.

4.1 OpenStack

OpenStack[3] is a open source community effort for an 'Infrastructure as a Service' (IaaS), released initially under the initiative of NASA[4] and RackSpace[5]. As of present, in its 11th release (Kilo), it features a rich catalogue for virtualizing different parts of a datacenter infrastructure.

In OpenStack, Nova provides compute services with support for a number of hypervisors from open source solutions such as KVM[6] or Xen[7] to commercial packages such as VMware[8]. Swift is the object storage solution with the ability to manage peta byte storage systems, while Cinder provides block storage to instances. Glance is the image repository with interface possibilities to different storage backends. Keystone is the primary secure access gateway to the cloud. It represents a unified identity management system handling users, groups and their permissions across all service offerings. The network is virtualized by Neutron, which allows the tenants to implement their own virtual networks. Horizon is the dashboard providing web-based user interface to OpenStack services. Heat implements the orchestration engine for cloud applications infrastructure deployment in an automated fashion. It has its own template language but also supports the AWS CloudFormation[9] template format. Ceilometer serves as the metering service for the deployed OpenStack infrastructure.

4.2 Gluster File System

GlusterFS[10] is a distributed file system for building storage solutions that are flexible, automated and adaptive to growing demands of data. GlusterFS has been designed to create high level abstractions on top of which storage infrastructures can be build to support new cloud computing applications or to match specific workload profiles. GlusterFS is deployed through the concepts of bricks and translators. Bricks, in their simplest form are servers attached to the network with a storage element that when stacked together, can implement the functions of RAID. The location of data is determined algorithmically, thus eliminating the need for centralized metadata stores.

3. *http://www.openstack.org*
4. *http://www.nasa.gov*
5. *http://www.rackspace.com*
6. *http://www.linux-kvm.org*
7. *http://www.xenproject.org*
8. *http://www.vmware.com*
9. *http://aws.amazon.com/cloudformation*
10. *http://www.gluster.org*

4.3 Host Identity Protocol

The Host Identity Protocol (HIP) [1, 9] introduces a cryptographic namespace used to identify end-hosts. Similarly to Secure Shell (SSH), the identifiers are essentially public keys that are compressed by hashing the public key. The namespace is compatible with existing applications by the use of virtual addresses. The namespace is managed by a new shim layer between transport and network layers that translates cryptographic identifiers into routable addresses. The protocol could be characterized as a VPN, but with the difference that it works without a gateway. Similarly to many other VPNs, HIP employs IPsec [10] to authenticate and encrypt application traffic.

HIP improves three different aspects in the way applications can address each other. First, it supports persistent identifiers in the sense that applications use virtual addresses that remain static despite a virtual machine using HIP would migrate to another network. The identifiers also remain consistent even in private address realms, which makes HIP a suitable solution for NAT traversal. Second, it supports heterogeneous addressing with both IPv4 and IPv6 being easily interchangeable. Third, the namespace in HIP is a secure environment with unique identifiers that can not be forged or 'stolen'. While a number of other protocols meet these constraints partially, HIP fulfils all three. Due to space constraints, the comparison of related protocols cannot be presented here, but is described in full detail in our earlier work [6, 7].

HIP achieves persistent identifiers by introducing a new namespace for the transport and application layers that is decoupled from the network layer addresses. The identities are managed by a new logical layer between the transport and network layers that manages the bindings between the identifiers and locators as illustrated in Figure 1.

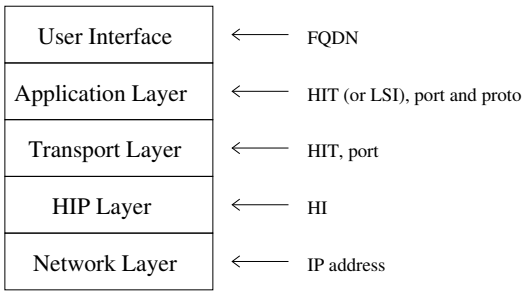


Fig. 1: HIP decouples the identifiers of the upper layers from the locators used at the network layer.

The new namespace is based on public-key cryptography. According to HIP terminology [9], an abstract identity is referred to as a Host Identity, whereas a Host Identifier (HI) refers to the concrete representation format of the corresponding identity, that is, the public key of a host. The end-host is responsible for creating the public key and the corresponding private key for itself. This way, a HI is self-certifying and statistically unique.

Public keys can be of variable length. Therefore, they are unsuitable to be used in fixed-length headers for the HIP control plane, and they are also incompatible with legacy IPv4 and IPv6 applications. To overcome this, a HIP-capable host creates two other compressed representations of the HI. The format for the control plane and IPv6 applications is the same: the host calculates a hash over the HI in order to fit it into an IPv6 address and sets a special prefix for the generated IPv6 address in order to distinguish it as a virtual address. For IPv4 applications, the host locally assigns a private IPv4 address, called a Local-Scope Identifier (LSI), that acts as an alias for the HI. This way, HIP can support unmodified legacy IPv4 and IPv6 applications. It should be noted that since the identifiers in HIP are not routable, the HIP layer translates them into routable addresses, or locators in HIP terms. The translation occurs within a new shim layer located between the transport and network layers.

HIP experiment report [11] describes the use of HIP with a number of applications. However, report does not mention anything on scientific applications.

4.4 ARC Middleware

The Advanced Resource Connector (ARC) [12] is a grid middleware stack developed by the NorduGrid collaboration[11]. ARC is widely deployed and used by research centers and institutes across the Europe and especially in the Nordic countries. The aim is to provide a component-based portable, compact and manageable grid middleware for the community. Along with other European middleware, ARC is also part of the European Middleware Initiative (EMI)[12] project.

Following are the key components of ARC software suite:

- **HED**: Hosting Environment Daemon (HED) is the web service container specialized for grid related functionality.
- **Computing Element (CE)**: ARC Resource-coupled Execution (A-REX) service provides the CE functionality. Other features include the support of Run Time Environments (RTE), logging capabilities, support of Local Resource Management Systems (LRMS) including SLURM[13], PBS[14], Condor[15].
- **Information System**: ARC resources can be discovered by using Information System Indexing System (ISIS) web service. ISIS services maintain a peer-to-peer network in order to provide fault-tolerance and high availability.
- **ARC UI**: ARC supports command-line and graphical user interface for job submission and management.

11. *http://www.nordugrid.org/arc/*
12. *http://www.eu-emi.eu*
13. *http://slurm.schedmd.com*
14. *http://www.pbsworks.com*
15. *http://research.cs.wisc.edu/htcondor/*

- **Caching Service**: This service is an optional part of the A-REX service. The aim is to reduce unnecessary data transfer by caching input data files for the subsequent jobs.
- **Data Management**: For data management, ARC relies on GridFTP server with some customizations.
- **Libraries and APIs**: ARC provides libraries that provide certificate handling, resource discovery, brokering and data management functionality.

4.5 CMS Analysis Environment

The Compact Muon Solenoid (CMS) collaboration consists of 182 institutes from 42 countries. CMS Monte Carlo simulation and physics analysis jobs can be submitted to the resources directly from ARC UI or from elsewhere in the WLCG[16] through EMI or the pilot workload management systems.

Computing jobs in High Energy Physics (HEP) data analysis and simulation are relatively simple and sequential. These jobs require a well defined and complex software environment. The base requirement for a CMS physics computing job in a cloud environment is to be able to run a correct VM. The VM should contain the correct runtime libraries and be able to install the correct analysis software. The method used here is CernVM File System (CVMFS)[17], a scalable software distribution service to assist HEP collaborations to deploy their software on VMs and other grid sites.

4.5.1 CMS Analysis Example

The CMS experiment is a multi-purpose detector to register signals from particle collisions at the LHC[18] at the European Organization for Nuclear Reseach (CERN)[19]. The recorded data are processed with applications built within the CMS Software Framework (CMSSW) [13].

Typical CMS data analysis jobs read reconstructed events that originate either from the particle collisions or from their simulation. The data is read directly from a server hosting the data files using the xrootd protocol [20]. The output of an analysis job is either a set of histograms of quantities of interest, or an interesting subset of the events in some reduced data format. Usually this is in the form of ROOT [14–16] objects, for subsequent fast analysis steps. Analysis jobs have a high I/O load, and the performance can be limited by it.

The grid jobs are managed with CMS Remote Analysis Builder (CRAB) [17], a tool developed within the CMS community to provide a common interface for communicating with different grid environments. The CRAB tool also automates the communication to the CMS Dataset Bookkeeping System (DBS) [18], i.e., searching and publication of new data files for jobs. A CRAB job consists of the following elements:

- a shell script setting up the CMSSW job environment, running the job, possibly copying the output file to a remote server, and cleaning up;
- CMSSW job configuration file;
- all libraries built on top of a release;

Especially the last item can pose problems in practice if the set of libraries is large ($\gtrsim 10\,\mathrm{MB}$) and the network bandwidth is limited between the machine where the grid jobs are submitted and the CE. This occurs because currently all the information is sent along each ARC job.

5 System Architecture

For our purpose, we have deployed a private cloud based on OpenStack software suite within the locally available resources at University of Helsinki, Computer Science Department[21], Finland. For peak hours, we have utilized 'Pouta' community IaaS cloud[22] from the IT Center for Science Ltd (CSC)[23], Finland. CSC provides computational, storage and network facilities for research and academic purposes in Finland.

The setup has been designed to run the CMS analysis framework, originally designed to run on the distributed infrastructures managed by grid technologies. Therefore, in order to be the part of the CMS community, we needed to expose DII-HEP cloud to various job submission and monitoring interfaces. Supporting the legacy components from the grid infrastructure while at the same time establishing a reliable, scalable and secure cloud infrastructure, our presented solution can be perceived as a grid-enabled, cloud-based computing cluster.

Our infrastructure presented in Figure 3, has a multi-server architecture with a clean separation of duties between management, computing, networking and storage. The technique we employ is to logically partition the architecture into multiple tightly interacting tiers. Each tier represent an abstraction of a resource provided with the capability to expand and contract upon demand.

In the private cloud, the physical storage tier offers plain storage Logical Unit Numbers (LUNs) through 10Gbit iSCSI interfaces. We configured two servers as GlusterFS storage bricks each equipped with 2TB LUNs. We created a mirrored shared volume where the compute nodes mount to in order to boot the virtual instances. For improved reliability and availability, we employed separate GlusterFS volumes for the MySQL database and the virtual machine image repository (Glance). GlusterFS peering and management is configured from the OpenStack controller as a logical entity in form of a GlusterFS admin node. The role of the GlusterFS daemon running in the Gluster admin node is to act as a control plane entity with functions of

16. *http://wlcg.web.cern.ch*
17. *http://cernvm.cern.ch*
18. *http://home.web.cern.ch/topics/large-hadron-collider*
19. *http://home.web.cern.ch*
20. *http://xrootd.org*
21. *http://www.cs.helsinki.fi*
22. *http://pouta.csc.fi*
23. *http://www.csc.fi*

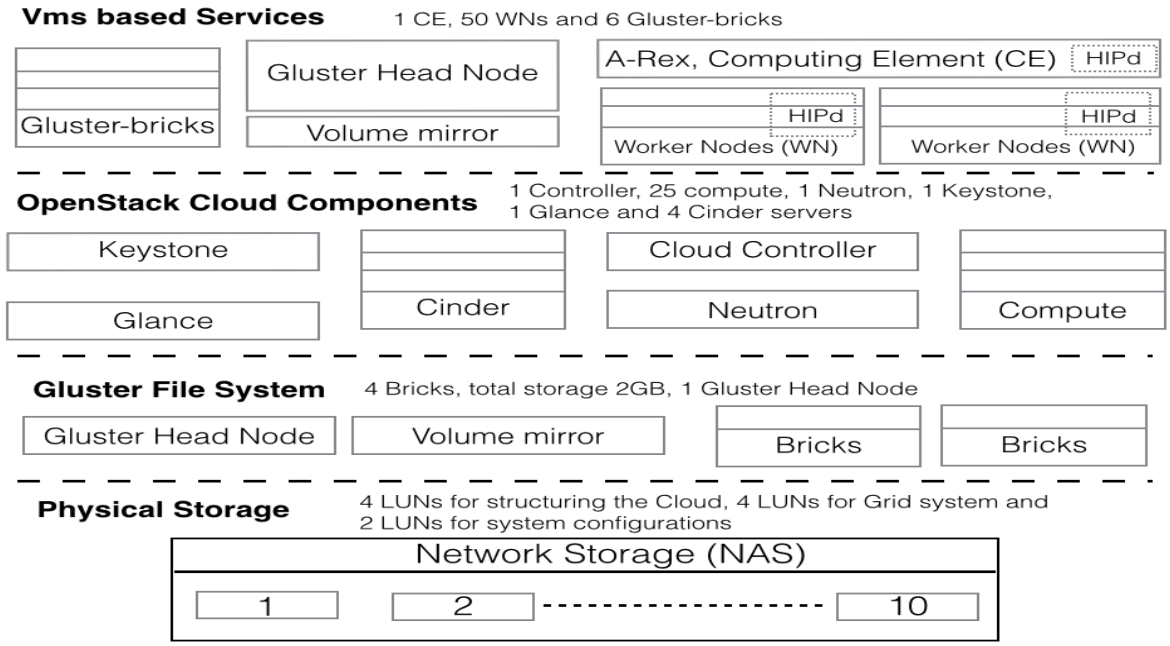


Fig. 2: Overall system architecture of DII-HEP cloud based on Openstack, GlusterFS and Grid tools.

updating, creating and enforcing policies to the volumes created.

The core cloud tier encompasses the key elements of the OpenStack suite. The cloud controller provides resource management and orchestrating activities by running: nova-api, nova-scheduler, nova-cert, nova-consoleauth, cinder-scheduler, Keystone, Glance and Horizon, but also the server instances of RabbitMQ, MySQL and GlusterFS. In addition to the cloud controller, a network controller manages the network services related to IP address management, DNS, DHCP and security groups through the elements of neutron-server, neutron-dhcp-agent, and neutron-l3-agent. At a high level, Neutron displays a very plug-in oriented architecture. Leveraging the concepts of linux network namespaces, it provides an isolated virtual networking stack for each tenant/user with its own network, interfaces, and routing tables.

GlusterFS is used as storage backend for running the block based storage provided by Cinder. Four of our server instances are configured to run as cinder volume nodes providing the storage volumes needed by the CMS analysis software.

The rest of the machines, referred to as Compute Nodes, run the nova-compute service which interacts with underlying hypervisor through the libvirt API. Each compute node has also installed a GlusterFS client that provides file system services for the unified mount point where the VMs boot at, and an OpenvSwitch instance that forwards the network traffic of the VMs running on the compute nodes.

The virtual environment tier provides the actual application infrastructure in the form of virtual instances that run a Compute Elemenent (CE) and number of Worker Nodes (WNs). Using the A-REX service provided by the ARC software stack, the CE manages the authentication of clients and distribution of jobs to the WNs. To fulfil the storage requirements of the application tier, the setup is supported by a GlusterFS distributed storage configuration with one admin node and six storage bricks. The CE is publicly exposed to acquire jobs from the grid interface. It should be noted that monitoring and accounting of the available resources is also one of the integral parts of running a CMS site. The ARC CE runs the information services that collect and submit site level information to the publicly available accounting and monitoring systems.

In our architecture, HIP is used to secure inter-VM communications with minimal deployment hurdles at the client side. Despite of how tenants in private clouds are isolated, we employ HIP mostly to harness its potential for portable and secure addressing for hybrid clouds. The proposed architecture allows the application experts to create HIP enabled secure environments within the application domain, while securing the computation and the data. It is worth highlighting that we do not yet utilize HIP in securing the VM communication with the storage elements as this remains future work.

OpenStack itself does not provide any built-in monitoring tools. Instead, it relies on integrating with other software that can be customized according to the requirements. We employed Graphite[24] and the *collectd* plugin[25] to build a centralized monitoring tool to evaluate the

24. *http://graphite.wikidot.com*
25. *http://collectd.org*

performance at three different levels. Level 1 monitors the CPU usage and the overhead induced by HIP/SSL, while levels 2 and 3 provide statistics on memory and network performance.

6 Hybrid Cloud Setup

With a hybrid cloud setup, organizations scale out the local infrastructure (often provided by a private cloud) by renting resources from publicly available clouds. For the scientific community, this model is cost efficient because it allows to start small with locally available resources and then later to expand temporarily in order to conduct experiments requiring larger computational capacity.

In hybrid cloud scenarios, the computational elements and data need to be exposed to Internet, thus network security is a major concern. As the tools and frameworks for cloud security are still maturing, more efforts are required to provide a comprehensive cloud based solution. We have herein focused on providing operating system level security mechanism based on HIP. As a major benefit no modifications are required at the application level.

We have prepared customized VM images to support HIP. The data transfer between the CE and the WNs is secured by HIP. The details regarding HIP-enabled instances are provided in Section 7. Similar to our DII-HEP private cloud infrastructure, we have launched several WNs on the CSC community cloud and connected them to the CE using HIP.

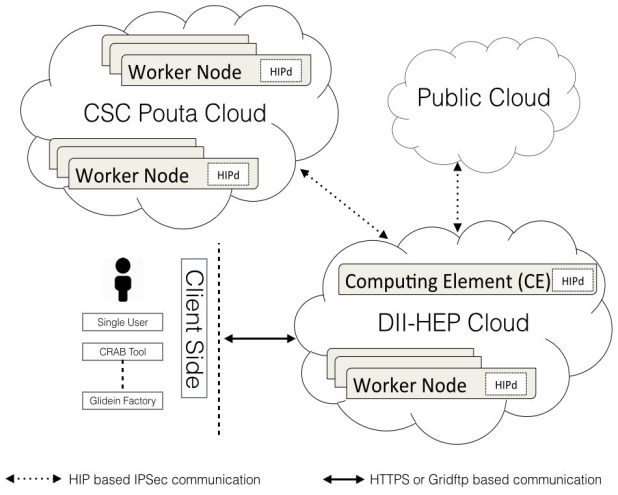


Fig. 3: Hybrid cloud environment based on DII-HEP and Pouta Cloud.

Here, it is important to note that our hybrid cloud solution does not require any extra components or mechanisms to manage public resources when compared to private ones. At present as a limitation, instance initialization at the public cloud is not dynamic but rather requires manual starting of VMs.

DII-HEP receives jobs coming from different client submission tools using HTTPS and GridFtp interfaces, while HIP is not involved at this stage. Once the jobs have been staged in, the CE further schedules these jobs to either run on the DII-HEP cloud or CSC cloud resources. From here onwards, all the communication is done using HIP that authenticates the CE and WN using their public keys. In addition it encapsulates the application payload into a secure IPsec tunnel. Once the job has been finished, the reverse path is followed to stage out the results.

Figure 3 illustrates our proposed model for hybrid clouds that we have also implemented. The solid line shows the communication channel secured by HTTPS and the dotted lines represent channels secured using HIP and IPsec. In our case, CSC cloud was our choice for scaling out infrastructure, but the current framework can also utilize resources from any public cloud that serves the customized VM images.

7 Experimental Setup

DII-HEP cloud deployment spans over 50 machines acquired from our private HPC cluster. Each machine is equipped with eight 2.53 GHz Intel Xeon CPU cores, 32 GB RAM and four 10 Gbit Network Interface Cards (NICs). The whole physical infrastructure is connected through Dell PowerConnect M8024 10GbE switches. With the availability of multiple NICs, we run separate networks for OpenStack administration, internal VM traffic and public Internet access. In addition to these, the machines that provide storage services have dedicated NICs for iSCSI communication with the storage array. We used Ubuntu 12.04 LTS for running the OpenStack installation ('Havana' version). The OpenStack installation uses KVM (Kernel Virtual Machine) as the underlying virtualization layer. In addition, each compute node is running an OpenvSwitch[26] instance that facilitates the implementation of virtual network abstractions by providing connectivity between VMs and physical interfaces. OpenvSwitch supports traditional switching techniques including the 802.1Q VLAN, QoS configurations and tunneling techniques such as generic routing encapsulation (GRE). In our setup, we employ GRE tunnels for traffic isolation between tenants.

At the application level, we run a CE virtual instance with 4 cores and 8 GB RAM. The WNs each have 4 cores, 14 GB RAM, and 40 GB of ephemeral disk space. GlusterFS provides the storage services, controlled by the admin node that has 4 cores and 12 GB RAM. The actual storage consists of six bricks each of which are equipped with 8 cores, 12 GB RAM and additional volumes of 500 GB. These bricks are replicated twice for fault tolerance, and they provide a distributed, but aggregated shared storage namespace that serves initially as the staging area for both new and finished

26. *http://openvswitch.org*

jobs. The ephemeral disks are configured locally for WNs that process the grid workloads. The VM images are relatively light weight (always smaller than 10 GB), which ensures that the images are rapid to deploy, terminate and migrate within the private or public cloud infrastructure. The VMs are based on Scientific Linux 6.4 distribution, with the latest grid software updates from the CERN repositories.

We use HIP to authenticate and secure the intra and inter-cloud communication between the CE and the WNs as highlighted with the dotted lines in Figure 3 and 4. The IPsec modules required for HIP are supported by the vanilla Linux kernel since version 2.6.27. At the first run of the VM, the HIP daemon generates its configuration files and its Host Identifier (i.e. the private-public key pair). As the scientific application is IPv6 capable, we employed HITs instead of the IPv4-compatible LSIs to enforce HIP-based communication. Only a single machine, the CE, initiates communication towards the WNs so we just stored the HIT-to-IP address mappings for HIP software locally in the */etc/hosts* file of CE instead of utilizing DNS [19] to store this information. It should be noted that we employed IPv4 as the routable locators because the used OpenStack version had some IPv6 issues, albeit the IPv6 is more mature in later releases.

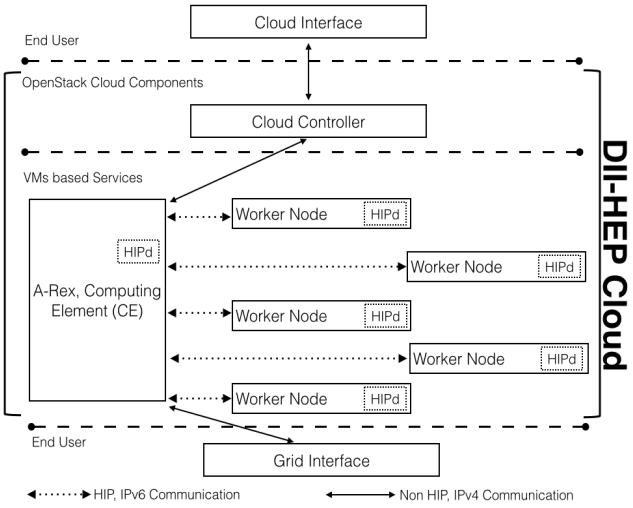


Fig. 4: HIP-based communication within the private DII-HEP cloud.

Facilitating the communication with the outside world, we expose two interfaces to the end users. The cloud interface is exposed via the cloud controller, which offers its service catalog endpoints for resources of compute, storage and networking. Each user with proper credentials can initiate, modify, or terminate the current setup accordingly. The grid interface is exposed through the CE entity. The interface provides all the necessary software tools required to interface with client-side GRID APIs. The interfaces and the communication with the storage elements inside the cloud environment where the WNs mount to read and write data are exposed through standard IPv4 features. All our instances are patched with preconfigured *collectd* daemon that reports to the *graphite* server. This enables us to monitor and collect performance data for the entire infrastructure.

8 Results and Discussion

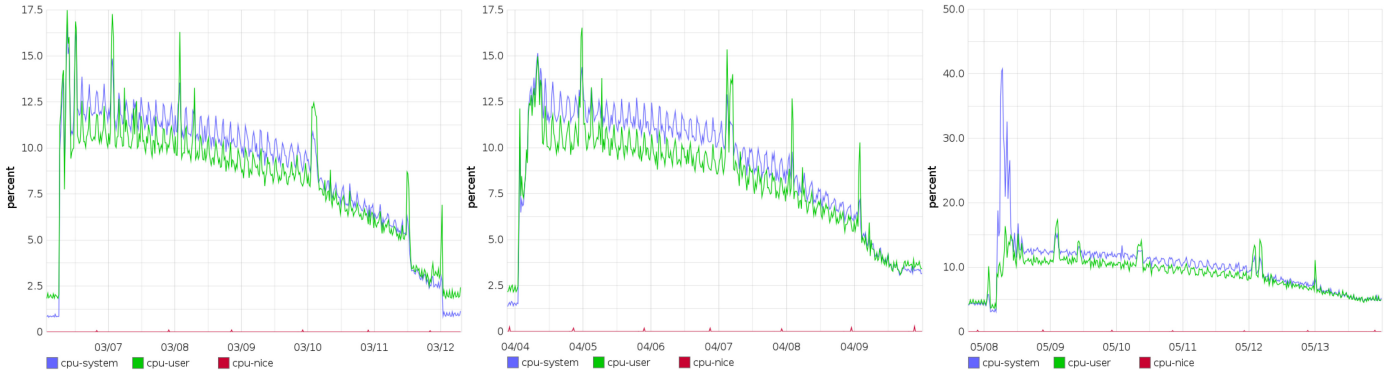
In order to evaluate the performance and stability of our infrastructure, we have conducted a series of experiments based on DII-HEP production cloud. The features of HIP as a protocol has been extensively evaluated and presented in articles cited in the subsection 4.3. Here our focus is completely on the application side. The experiments were conducted using the CMS standard grid job submission tool (crab) with a typical physics analysis jobs. The runtime performance of the jobs was similar to another CMS production site. Currently most of production sites for CMS analysis run jobs on physical nodes. Therefore, it is important to determine the performance degradation caused by a virtualized CMS. The detailed results concerning the execution in virtual environment are presented in [20]. We have measured only **4%** CPU degradation by running the HEP-SPEC06 benchmarks using VMs when compared to physical nodes [20]. This section describes further performance measurements that compare the overhead of different security mechanisms on our private cloud. The measurements in the public cloud are omitted due to space restrictions and similarity with the private cloud results. High-Performance Computing (HPC) applications have also been measured in public clouds by others [21–23].

It should be noted that two aspects of OpenStack are not measured here as others have already published results. Namely, Yang et al have measured VM migration performance [24], and Callegati et al have measured multi-tenancy aspects [25].

8.1 Application Performance Measurements

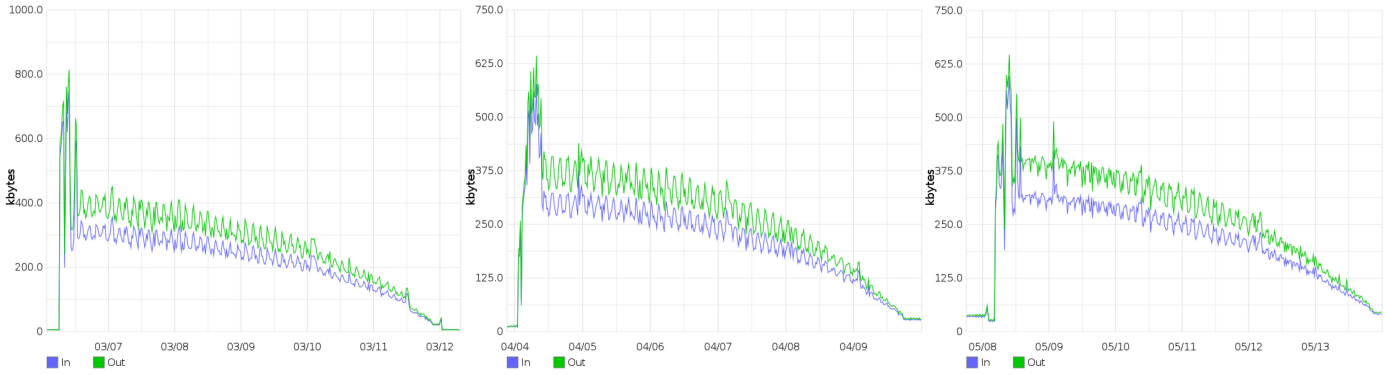
This section presents the insight of our implemented architecture in a real world production environment. The set of jobs submitted to the DII-HEP cloud, model fairly well the CPU and I/O load of a computationally intensive analysis job in CMS. We submitted and managed jobs with CRAB using the grid interface as shown in Figure 4.

Our tests focused on system performance to evaluate the overhead introduced by the HIP-based security mechanism. For this, we conducted three similar tests with three different security settings for the CMS analysis, each consisting of ten thousand grid jobs. The first test case uses no specific security mechanism, the second case uses HIP, and third one uses TLS/SSL based security. We compared the HIP and TLS/SSL cases with the insecure case as a baseline. Running a CMS job on a site requires users to have a valid X.509 certificate. Therefore the user must be a part of the authorized virtual organization [26].



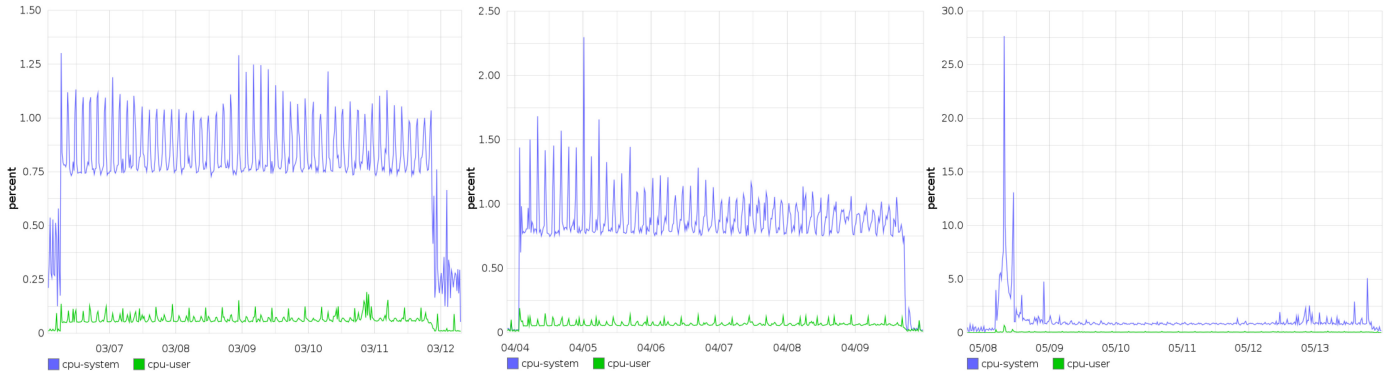
(a) Normal environment (b) HIP environment (c) SSL environment

Fig. 5: CPU utilization of the CE.



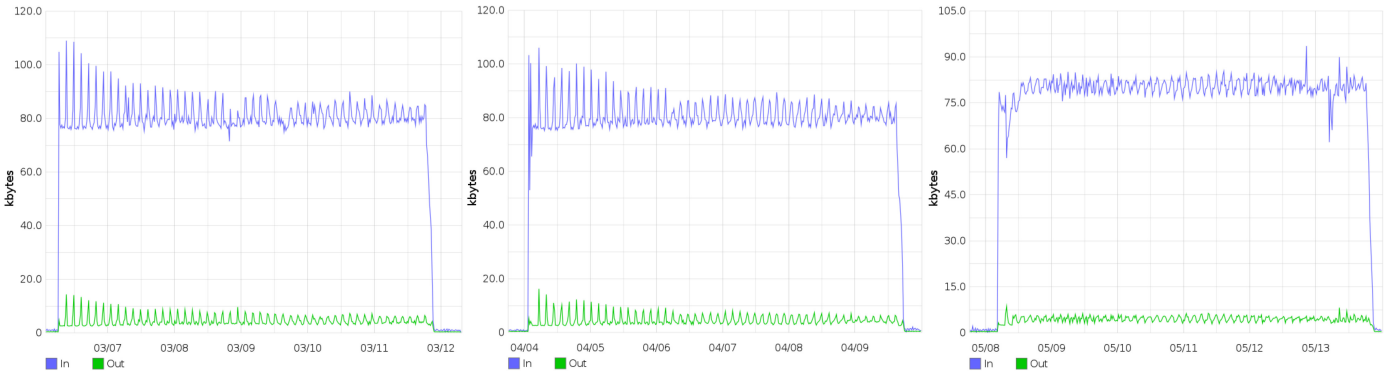
(a) Normal environment (b) HIP environment (c) SSL environment

Fig. 6: Network utilization of the CE.



(a) Normal environment (b) HIP environment (c) SSL environment

Fig. 7: Summed CPU utilization of the worker nodes by user and system processes only.



(a) Normal environment (b) HIP environment (c) SSL environment

Fig. 8: Summed Network utilization of the worker nodes.

Application level grid security modules impose this requirement in validating the incoming jobs, thus it is introduced as additional security measure in all the three test cases. In all three test cases, each grid job is autonomous and requires a serial execution of approximately **170** minutes. Each test case was running approximately for one week. The complete life cycle of a job includes submission, execution and data staging. The jobs are CPU bound, and require access to parts of the event data organized as files on the common storage sites using the *xrootd* protocol. The success rate in all three test cases is greater than **99%**, with some jobs failing due to data staging issues. All three test cases were executed by a cluster of **50 WNs**, each equipped with four cores. In the grid execution environment, each core at a WN was configured as a single job execution slot. In total, 50 WNs contributed 200 slots for parallel job execution. Also, each slot can consume up to 1 GB of resident memory.

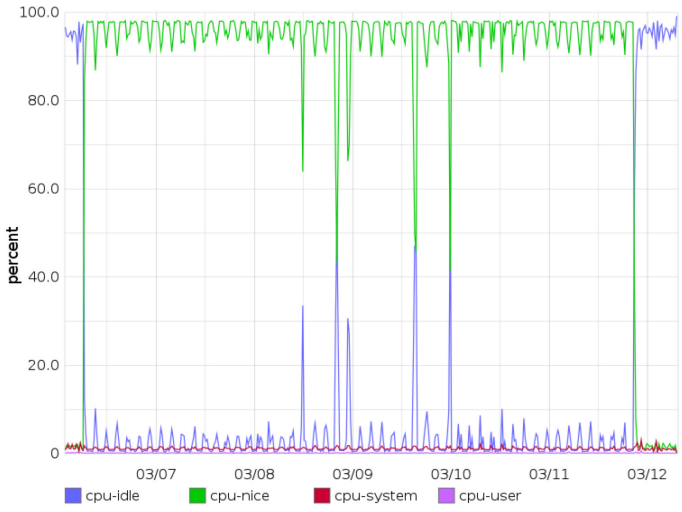


Fig. 9: Utilization of a single core in a worker node.

Figure 9 illustrates the utilization pattern of a single-core CPU in a WN. We have used the baseline (no HIP or TLS/SSL security) test case to generate this plot. The X-axis shows number of days the test was running and the Y-axis the CPU utilization in percentage. The CPU activity shown by the *CPU-user* is based on the percentage of the user processes, e.g.applications, whereas *CPU-nice* reflects the percentage of CPU occupied by the user processes (applications) with positive nice value. Since the WNs are dedicated to run the CMS jobs, the single-core execution shows that the CPU core is mostly occupied by the CMS jobs. The CPU utilization by the remaining processes is negligible in comparison to the CMS job processes. We observed a similar pattern while running the other two test cases. It is worth noting that the green line (*CPU-nice*) dips 50 times, each signifying a separate job running for roughly the same time. The same trend occurs also in all three test cases.

Figures 5, 6, 7 and 8 show the CPU utilization and network traffic of the CE and WNs in the three test cases. In each figure, the first subfigure shows the system performance without security, the second one illustrates the HIP case and the third one shows the TLS/SSL case. It is worth noting that the scale in the TLS/SSL case is different due to occasional peaks. The plots in Figure 5 show that the overall CPU load in each of the three test cases is less than **20%** at the CE. All three cases also experience a very similar performance overhead except for a few peaks in the SSL case. This shows that including HIP-based security creates negligible CPU overhead at the CE. The same behavior can also be observed in the network performance as shown in Figure 6. Figures 7 and 8 show the summed CPU and network usage for the 50 WNs. Here it is worth noting that SSL operates in the user space whereas IPsec operates in the kernel. All three cases incur similar performance overhead, and we have also excluded *CPU-nice* in Figure 7 for improved clarity.

The overhead of network security appears to be relatively small with the workloads we have used in our experiments. The reason for this is that the workloads are not network intensive, and thus security is not the main bottleneck. In next section, we stress test the network with different security solutions.

8.2 Network Performance Comparison

We measured raw TCP throughput and network latency (ICMP round-trip time, RTT) using different security protocols installed on the VMs. For these measurements, we used hardware different from the other measurements; the rack servers consisted of HP Proliant BL280c server (2 Intel Xeon CPU E5640 2.67Ghz, 64 GB of memory) with Gigabit ethernet interfaces. The OpenStack release was Grizzly, and Ubuntu 12.04 was used in the VMs. The results are shown in Figure 10.

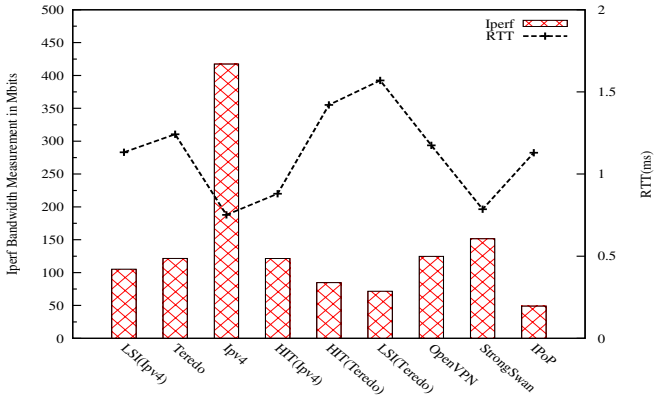


Fig. 10: Network throughput and latency.

As the base case, plain unsecured IPv4 connectivity (marked with "Ipv4") displays best performance as it has the highest throughput (417.4 Mbit/s) and smallest latency (0.8 ms). For HIP measurements, we employed HIP for Linux v1.0.7. HIP was measured with different combinations of application layer identifiers (LSIs vs. HITs)

and network layer locators (IPv4 vs. IPv6 vs. Teredo), of which HITs combined with IPv4 based locators offered the best performance. This measurement result will be used for comparing with other VPNs (StrongSWAN v5.0.4, OpenVPN v2.2.1, iPoP v9.6.1) because the same HIP configuration was used in the grid measurements.

In network throughput, StrongSWAN offered the best performance (151.5 Mbit/s). OpenVPN (124.7 Mbit/s) and HIP (121.5 Mbit/s) offered very similar performance. iPoP displayed the worst performance (49.0 Mbit/s).

In network latency, StrongSWAN dominated again (0.8 ms), but HIP came second (0.9 ms) and then iPoP (1.1 ms). The worst performance occurred with OpenVPN (1.2 ms).

To generalize, VPN-based security introduces a large performance penalty to the throughput, but smaller impact to the latency. In other words, VPN-based security should be avoided in high-throughput applications when performance is critical. Of the four compared VPNs, the chosen protocol, HIP, offers "medium" performance.

9 A Straw-man Security Analysis

In this section, we review some of the security properties of the components based on the categorization of the classic Confidentiality, Integrity and Availability (CIA) information security model [27].

9.1 Confidentiality and Integrity

In our specific use case, the data is not highly sensitive. The entire CMS community can have read access to the CMS data and write back the computed results. It is enough to keep outsiders outside, and we have no need for multiple access levels to different types of data. At the application level, virtual organization is enough to keep outsiders out. However, the multi-tenant environment as introduced by public clouds requires additional isolation mechanisms below the application layer. Within a single cloud, the cloud vendor should employ hypervisors to guarantee confidentiality and integrity protection for virtualized applications and their data. In our case, we employed the KVM hypervisor. When the data is being moved within a single cloud, vendor specific isolation guarantees, such GRE tunnels in the case of our OpenStack deployment, can be used as a security measure. Finally, we proposed to use tenant-specific measures to secure hybrid cloud scenarios in order to support "portable" and vendor-agnostic security. More specifically, we employed HIP-based VPN connectivity directly between virtual machines in our deployment to provide authenticity, integrity and confidentially protection using IPSec.

It is worth noting that our threat model assumes that a public cloud vendor does not maliciously compromise credentials related to the virtual organization, OpenStack authentication or even the private keys used for HIP. We believe this is not an unreasonable assumption because such operations would hurt economically the public cloud vendor in the long run, and needs to be agreed clearly in the service-level agreement with the vendor.

9.2 Availability

The ARC application we have employed supports fault-tolerance and recovery against benign problems, such as programming errors in the grid application or hardware failures. In such events, the CE can resubmit jobs to working WNs. While the system supports multiple CEs possibly even sharing WNs, the CE itself is the entrance point into the system and thus could be considered as the weakest link of the scientific application due to non-distributed nature. Fortunately, mounting, e.g., a Distributed Denial of Service (DDoS) attack to a CE outside of the cloud can be difficult due to two levels of network security, i.e., HIP tunnels transported inside OpenStack GRE tunnels.

A small private cloud deployment could still become a victim of a DDoS attack. In such a case, scaling out to a public cloud would make mounting such an attack harder due to larger attack surface, especially if the public cloud vendor operates on multiple continents.

10 Conclusion and Future Directions

We have virtualized a fairly complex scientific application with KVM and OpenStack. As the application is virtualized to run in the cloud, it is easier to deploy and scale out especially in public clouds. Private clouds are still of equal importance especially when dealing privacy-sensitive data. Mixing of private and public clouds in the form of hybrid clouds is becoming relevant; some organizations start with small private clouds, and scale out temporarily to public clouds to handle peak loads. However, hybrid cloud scenarios require additional mechanisms to deal with network topology changes and security. As a solution, we propose to use the Host Identity Protocol (HIP) for scientific applications operated in hybrid clouds.

Our network performance measurements show that HIP offers comparable performance to other three VPN technologies with some variations. In general, all four VPN technologies introduce a large network performance penalty on throughput when compared to insecure communications, but the impact on latency is negligible. Next, we measured and compared secured versions of the scientific application with a non-secured one. The results show that SSL/TLS or HIP as a security measure do not drastically impact the performance footprint in production environment. At first, this seems counter-intuitive because the pure network performance of TCP/IP degrades a roughly factor of five for throughput and a factor of two for latency when HIP is used in the OpenStack network. However, the unexpectedly low impact of the extra security in the production environment can be explained by the relatively low volumes

of network traffic in the grid. In environments with multiple tenants running workloads simultaneously, the additional security measures can have more impact. While others have measured the impact of multi-tenancy with OpenStack [25], this remains further work with our use case and application. We should also note that performance impact of HIP is a trade-off towards secured, vendor-agnostic and topology-independent addressing.

For the future, we are aiming to make our cloud more resilient, elastic and manageable at large scale without compromising security. For this we are currently proceeding in three directions. Firstly, we are working on dynamic provisioning of resources in hybrid cloud environments. Secondly, the live VM migration support in OpenStack has been improving over the time, and we would like to measure it using different mobility management schemes, albeit cross-domain migration appears still challenging. Thirdly, light-weight containers, such as those popularized by Docker, offer faster boot-up time for virtual machines, which we could employ to accommodate peak loads faster in scientific applications.

Acknowledgments

The authors would like to thank CSC technical staff and CMS collaboration for their support, and Mohit Sethi, Ramasivakarthik Mallavarapu, Heikki Oirola, Vivek Balakrishnan, Juhani Toivonen and Jimmy Kjällman for their help with this publication. The presented work is funded by Academy of Finland, grant numbers 255932 (CS) and 255941 (Physics).

References

[1] R. Moskowitz, T. Heer, P. Jokela, and T. Henderson, "Host Identity Protocol Version 2 (HIPv2)," RFC 7401 (Proposed Standard), Internet Engineering Task Force, Apr. 2015. [Online]. Available: http://www.ietf.org/rfc/rfc7401.txt

[2] R. A. Popa, E. Stark, J. Helfer, S. Valdez, N. Zeldovich, M. F. Kaashoek, and H. Balakrishnan, "Building web applications on top of encrypted data using mylar," in *USENIX Symposium of Networked Systems Design and Implementation*, 2014.

[3] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "Cryptdb: Protecting confidentiality with encrypted query processing," in *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, ser. SOSP '11. New York, NY, USA: ACM, 2011, pp. 85–100. [Online]. Available: http://doi.acm.org/10.1145/2043556.2043566

[4] M. van Dijk and A. Juels, "On the impossibility of cryptography alone for privacy-preserving cloud computing," Cryptology ePrint Archive, Report 2010/305, 2010.

[5] A. N. Toosi, R. N. Calheiros, and R. Buyya, "Interconnected cloud computing environments: Challenges, taxonomy, and survey," *ACM Comput. Surv.*, vol. 47, no. 1, pp. 7:1–7:47, May 2014. [Online]. Available: http://doi.acm.org/10.1145/2593512

[6] M. Komu, M. Sethi, and N. Beijar, "A survey of identifierâĂŞlocator split addressing architectures," *Computer Science Review*, no. 0, pp. –, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1574013715000040

[7] M. Komu, "A consolidated namespace for network applications, developers, administrators and users," Dec. 2012, iSBN 978-952-60-4905-2.

[8] A. Pathak, M. Komu, and A. Gurtov, "Host Identity Protocol for Linux," in *Linux Journal*, Nov. 2009. [Online]. Available: http://www.linuxjournal.com/article/9129

[9] R. Moskowitz and P. Nikander, "Host Identity Protocol (HIP) Architecture," RFC 4423 (Informational), Internet Engineering Task Force, May 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4423.txt

[10] P. Jokela, R. Moskowitz, and P. Nikander, "Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP)," RFC 5202 (Experimental), Internet Engineering Task Force, Apr. 2008, obsoleted by RFC 7402. [Online]. Available: http://www.ietf.org/rfc/rfc5202.txt

[11] T. Henderson and A. Gurtov, "The Host Identity Protocol (HIP) Experiment Report," RFC 6538 (Informational), Internet Engineering Task Force, Mar. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6538.txt

[12] M. Ellert *et al.*, "Advanced Resource Connector middleware for lightweight computational Grids," *Future Generation Comput. Syst.*, vol. 23, no. 2, pp. 219–240, 2007.

[13] C. D. Jones, M. Paterno, J. Kowalkowski, L. Sexton-Kennedy, and W. Tanenbaum, "The new CMS event data model and framework," in *Proceedings of International Conference on Computing in High Energy and Nuclear Physics (CHEP06)*, 2006. [Online]. Available: http://www.tifr.res.in/~chep06/

[14] R. Brun and F. Rademakers, "ROOT — an object oriented data analysis framework," *Nucl. Instr. and Meth. A*, vol. 389, no. 1–2, pp. 81–86, 1997.

[15] I. Antcheva *et al.*, "ROOT — a C++ framework for petabyte data storage, statistical analysis and visualization," *Comput. Phys. Commun.*, vol. 180, no. 12, pp. 2499–2512, 2009.

[16] "ROOT — a data analysis framework," accessed in Jan 2015. [Online]. Available: http://root.cern.ch/

[17] A. Fanfani *et al.*, "Distributed analysis in CMS," *J. Grid Comput.*, vol. 8, no. 2, pp. 159–179, 2010.

[18] M. Giffels *et al.*, "Data bookkeeping service 3 - providing event metadata in CMS," *J. Phys.: Conf. Ser.*, vol. 513, no. 4, p. 0420222, 2014.

[19] P. Nikander and J. Laganier, "Host Identity Protocol (HIP) Domain Name System (DNS) Extensions," RFC 5205 (Experimental), Internet Engineering Task Force, Apr. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5205.txt

[20] S. Toor, L. Osmani, P. Eerola, O. Kraemer, T. Lindén, S. Tarkoma, and J. White, "A scalable infrastructure for cms data analysis based on openstack cloud and gluster file system," *Journal of Physics: Conference Series*, vol. 513, no. 6, p. 062047, 2014.

[21] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "A performance analysis of ec2 cloud computing services for scientific computing," in *Cloud Computing*, ser. Lecture Notes of the Institute for Computer Sciences, D. Avresky, M. Diaz, A. Bode, B. Ciciani, and E. Dekel, Eds. Springer Berlin Heidelberg, 2010, vol. 34, pp. 115–131. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-12636-9_9

[22] K. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. J. Wasserman, and N. Wright, "Performance analysis of high performance computing applications on the amazon web services cloud," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, Nov 2010, pp. 159–168.

[23] B. Amedro, F. Baude, F. Huet, and E. Mathias, "Combining Grid and Cloud Resources by Use of Middleware for SPMD Application," in *2nd International Conference on Cloud Computing Technology and Science*, Indianapolis, IN, United States, Nov. 2010, pp. 177–184. [Online]. Available: https://hal.inria.fr/inria-00538549

[24] C.-T. Yang, Y.-T. Liu, J.-C. Liu, C.-L. Chuang, and F.-C. Jiang, "Implementation of a cloud iaas with dynamic resource allocation method using openstack," in *Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2013 International Conference on*, Dec 2013, pp. 71–78.

[25] F. Callegati, W. Cerroni, C. Contoli, and G. Santandrea, "Performance of multi-tenant virtual networks in openstack-based cloud infrastructures," in *Globecom Workshops (GC Wkshps), 2014*, Dec 2014, pp. 81–85.

[26] I. Foster, "The anatomy of the grid: Enabling scalable virtual organizations," in *Euro-Par 2001 Parallel Processing*, ser. LN in Computer Science, R. Sakellariou, J. Gurd, L. Freeman, and J. Keane, Eds. Springer Berlin Heidelberg, 2001, vol. 2150, pp. 1–4. [Online]. Available: http://dx.doi.org/10.1007/3-540-44681-8_1

[27] Y. Cherdantseva and J. Hilton, "A reference model of information assurance amp; security," in *Availability, Reliability and Security (ARES), 2013 Eighth International Conference on*, Sept 2013, pp. 546–555.

Lirim Osmani is a Ph.D. canditate in Computer Science at University of Helsinki, Finland. He is currently conducting research on datacenter management, optimization and virtualization technologies for intensive on-demand computing. He has industry engineering background in virtualizing enterprise systems.

Salman Toor has done his M.Sc. and Ph.D. in Scientific Computing from Uppsala University, Sweden. In 2012, Toor was hired as a post doctoral researcher in CMS program, Helsinki Institute of Physics, Finland. His area of expertise include management, scalability and performance of distributed computational and storage infrastructures for scientific applications. Currently have a split position at Uppsala University, employed as a researcher at Department of Information Technology and as cloud application expert at UPPMAX HPC Centre.

Miika Komu is working on cloud technologies at Ericsson Research in Finland after finishing his doctoral studies at Aalto University in Finland in 2012. Earlier, he has been working for Helsinki Institute for Information Technology and Aalto University. Besides research and teaching activities, he has been also involved with standardization activities in Internet Engineering Task Force.

Matti J. Kortelainen is a post-doc scientist at Helsinki Institute of Physics, working on the CMS experiment. He got his Ph.D. in physics from University of Helsinki in 2013.

Tomas Lindén received his Ph.D. in Experimental Particle Physics from University of Helsinki, Finland. Lindén is working as a senior scientist at the Helsinki Institute of Physics (HIP), coordinating the Tier-2 Operations project of the HIP CMS research programme. His area of expertise ranges from elementary particle physics to design and construction of large-scale distributed computing environments based on grid technologies.

John White received his Ph.D. in Physics in 1998 from the University of Victoria, Canada. Since he has worked at CERN for Carleton University and Helsinki Institute of Physics. He has worked in the European Grid projects in the security area and was the middleware security manager for the European Middleware Initiative and currently works on combining Grid and Cloud security.

Rasib Khan is a member of SECRETLab and a Graduate Research Assistant at the University of Alabama at Birmingham, USA. Khan was a NordSecMob EU Erasmus Mundus Scholar, and received his M.Sc degree from Royal Institute of Technology (KTH), Sweden, and Aalto University, Finland in 2011. His primary research interests include security technologies in identity management, cloud computing, and information provenance.

Paula Eerola is a professor at the Department of Physics, University of Helsinki, Finland. Her research field is elementary particle physics. She is leading the Finnish research programme at the CMS particle physics experiment at CERN, the European Organization for Nuclear Research, in Geneva, Switzerland.

Sasu Tarkoma received his M.Sc. and Ph.D. degrees in Computer Science from the University of Helsinki, Department of Computer Science. He is full Professor at University of Helsinki, Department of Computer Science and Deputy Head of the Department. He has managed and participated in national and international research projects at the University of Helsinki, Aalto University, and Helsinki Institute for Information Technology (HIIT). His interests include mobile computing, Internet technologies and middleware. He is Senior Member of IEEE.